

# Reconstructing Householder Vectors from TSQR

**Grey Ballard**, James Demmel, Laura Grigori,  
Mathias Jacquelin, Hong Diep Nguyen and Edgar Solomonik

SIAM Conference on Parallel Processing and Scientific Computing

February 19, 2014



- Householder QR is not fast enough for tall-skinny matrices
  - blocked algorithms can be bottlenecked by panel factorizations
  - applying (aggregated) Householder vectors = matrix multiplication
- Tall-Skinny QR (TSQR) [DGHL12] is faster for tall-skinny matrices
  - applying the implicit orthogonal matrix is more complicated
- We can get the best of both worlds at little extra cost
  - use TSQR but reconstruct the Householder vector representation
  - good for performance and software engineering

# Key Idea

Compute a QR decomposition  
using Householder vectors\*:

$$A = QR = (I - YTY_1^T)R$$



\* $I - YTY_1^T$  known as compact WY representation

# Key Idea

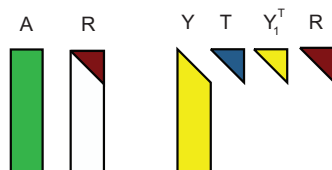
Compute a QR decomposition using Householder vectors\*:

$$A = QR = (I - YTY_1^T)R$$



Re-arrange the equation and we have an LU decomposition:

$$A - R = Y \cdot (-TY_1^T R)$$

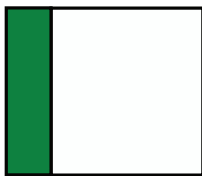


\* $I - YTY_1^T$  known as compact WY representation

# Householder QR (HhQR)

Blocked Householder QR works by repeating:

- 1 panel factorization (tall-skinny QR decomposition)
- 2 trailing matrix update (application of orthogonal factor)



Householder vectors computed  
and applied one at a time

$$I - \tau yy^T$$

(two parallel reductions per column)

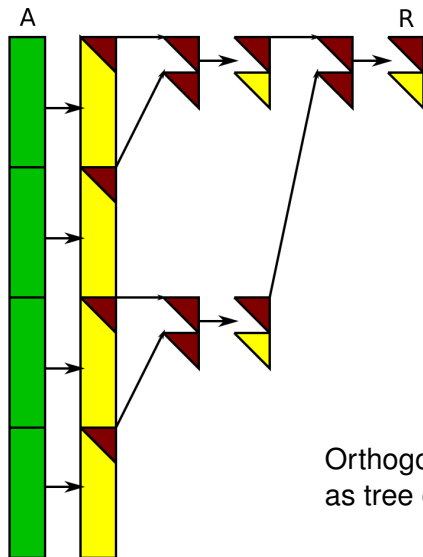


Householder vectors aggregated  
by computing triangular matrix  $T$

$$I - YTY^T$$

(application = matrix multiplications)

# Tall-Skinny QR (TSQR)

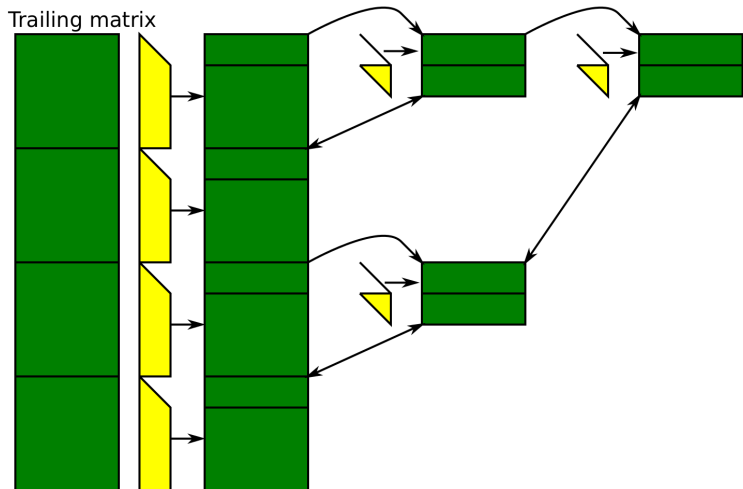


Key benefit of TSQR:  
one parallel reduction

Orthogonal factor stored implicitly  
as tree of Householder vectors

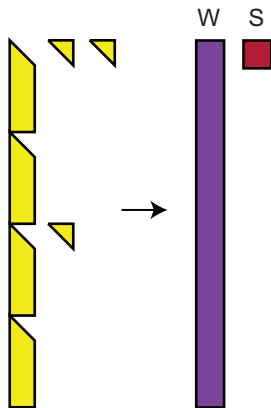
# Communication-Avoiding QR (CAQR)

CAQR uses TSQR for panel factorization and applies the update using implicit tree structure



# Yamamoto's Idea

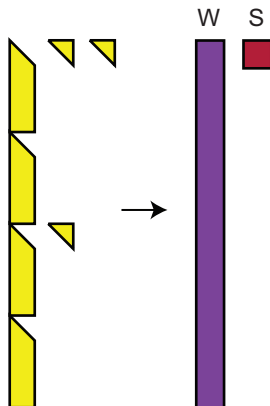
- Y. Yamamoto gave a talk at SIAM ALA 2012: he wanted to use TSQR but offload the trailing matrix update to a GPU
- To make CAQR's trailing matrix update more like matrix multiplication, his idea is to convert implicit tree into compact WY-like representation



# Yamamoto's Idea

- Y. Yamamoto gave a talk at SIAM ALA 2012: he wanted to use TSQR but offload the trailing matrix update to a GPU
- To make CAQR's trailing matrix update more like matrix multiplication, his idea is to convert implicit tree into compact WY-like representation

**Compact WY representation:**  $I - YTY^T$



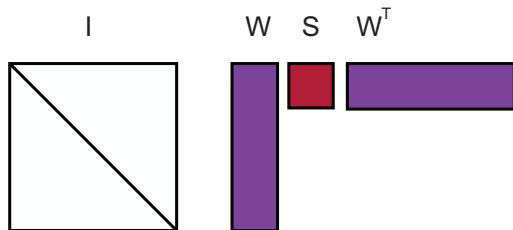
**Basis-kernel representation:**  $I - WSW^T$



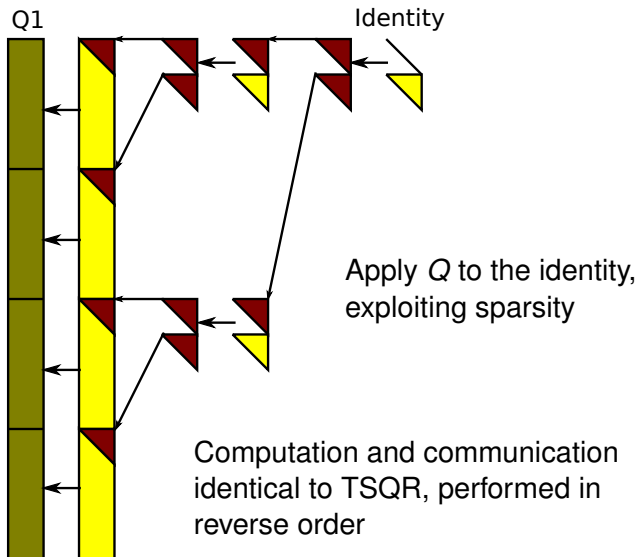
# Yamamoto's Algorithm

- 1 Perform TSQR
- 2 Form  $Q$  explicitly (tall-skinny orthonormal factor)
- 3 Set  $W = Q - I$
- 4 Set  $S = (I - Q_1)^{-1}$

$$I - WSW^T = I - \begin{bmatrix} Q_1 - I \\ Q_2 \end{bmatrix} [I - Q_1]^{-1} [(Q_1 - I)^T \quad Q_2^T]$$



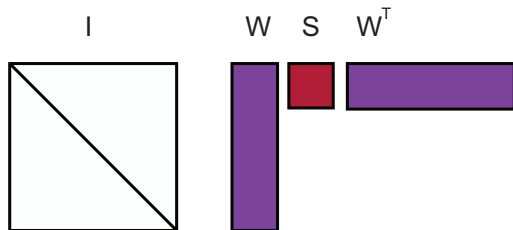
# How is $Q$ formed?



# Yamamoto's Algorithm

- 1 Perform TSQR
- 2 Form  $Q$  explicitly (tall-skinny orthonormal factor)
- 3 Set  $W = Q - I$
- 4 Set  $S = (I - Q_1)^{-1}$

$$I - WSW^T = I - \begin{bmatrix} Q_1 - I \\ Q_2 \end{bmatrix} [I - Q_1]^{-1} [(Q_1 - I)^T \quad Q_2^T]$$

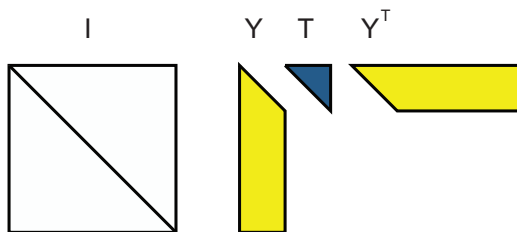


# Reconstructing Householder Vectors (TSQR-HR)

With a little more effort, we can obtain the compact WY representation:

- 1 Perform TSQR
- 2 Form  $Q$  explicitly (tall-skinny orthonormal factor)
- 3 Perform LU decomposition:  $Q - I = LU$
- 4 Set  $Y = L$
- 5 Set  $T = -UY_1^{-T}$

$$I - YTY^T = I - \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \begin{bmatrix} T \end{bmatrix} \begin{bmatrix} Y_1^T & Y_2^T \end{bmatrix}$$



# Why form $Q$ ?

Cheaper approach based on  $A - R = Y \cdot (-TY_1^T R)$ :

- 1 Perform TSQR
- 2 Perform LU decomposition:  $A - R = LU$
- 3 Set  $Y = L$
- 4 Set  $T = -UR^{-1}Y_1^{-T}$  (or compute  $T$  from  $Y$ )

# Why form $Q$ ?

Cheaper approach based on  $A - R = Y \cdot (-TY_1^T R)$ :

- 1 Perform TSQR
- 2 Perform LU decomposition:  $A - R = LU$
- 3 Set  $Y = L$
- 4 Set  $T = -UR^{-1}Y_1^{-T}$  (or compute  $T$  from  $Y$ )

This approach is similar to computing  $R$  using TSQR and  $Q$  using Householder QR

- if  $A$  is well-conditioned, works fine
- if  $A$  is low-rank, QR decomposition is not unique
- if  $A$  is ill-conditioned,  $R$  matrix is sensitive to roundoff

# What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition:  $Q - I = LU$ 
  - what if  $Q - I$  is singular?

# What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition:  $Q - I = LU$ 
  - what if  $Q - I$  is singular?

Actually, we need to make a sign choice:

- Perform LU decomposition:  $Q - Sgn = LU$ 
  - $Sgn$  matrix corresponds to sign choice in Householder QR
  - guarantees  $Q - Sgn$  is nonsingular
  - guarantees maximum element on the diagonal (no pivoting)

# What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition:  $Q - I = LU$ 
  - what if  $Q - I$  is singular?

Actually, we need to make a sign choice:

- Perform LU decomposition:  $Q - Sgn = LU$ 
  - $Sgn$  matrix corresponds to sign choice in Householder QR
  - guarantees  $Q - Sgn$  is nonsingular
  - guarantees maximum element on the diagonal (no pivoting)

No pivoting makes LU of tall-skinny matrix very easy

- LU of top block followed by triangular solve for all other rows

# Costs of Householder Reconstruction

## Householder Reconstruction

Let  $A$  be  $n \times b$

- 1 Perform TSQR  $2nb^2$  flops, one QR reduction of size  $b^2/2$
- 2 Form  $Q$   $2nb^2$  flops, one QR reduction of size  $b^2/2$
- 3  $LU(Q - Sgn)$   $nb^2$  flops, one broadcast of size  $b^2/2$
- 4 Set  $Y = L$
- 5 Set  $T = -U \cdot Sgn \cdot Y_1^{-T}$   $O(b^3)$  flops

# Costs of Householder Reconstruction

## Householder Reconstruction

Let  $A$  be  $n \times b$

- 1 Perform TSQR  $2nb^2$  flops, one QR reduction of size  $b^2/2$
- 2 Form  $Q$   $2nb^2$  flops, one QR reduction of size  $b^2/2$
- 3  $LU(Q - Sgn)$   $nb^2$  flops, one broadcast of size  $b^2/2$
- 4 Set  $Y = L$
- 5 Set  $T = -U \cdot Sgn \cdot Y_1^{-T}$   $O(b^3)$  flops

## Alternative Algorithms

- TSQR  $2nb^2$  flops, one QR reduction of size  $b^2/2$
- HhQR (and form  $T$ )  $3nb^2$  flops,  $2b$  reductions of size  $O(b)$
- Yamamoto's  $4nb^2$  flops, two QR reductions of size  $b^2/2$

# Costs of Householder Reconstruction

## Householder Reconstruction

Let  $A$  be  $n \times b$

- 1 Perform TSQR  $2nb^2$  flops, one QR reduction of size  $b^2/2$
- 2 Form  $Q$   $2nb^2$  flops, one QR reduction of size  $b^2/2$
- 3  $LU(Q - Sgn)$   $nb^2$  flops, one broadcast of size  $b^2/2$
- 4 Set  $Y = L$
- 5 Set  $T = -U \cdot Sgn \cdot Y_1^{-T}$   $O(b^3)$  flops

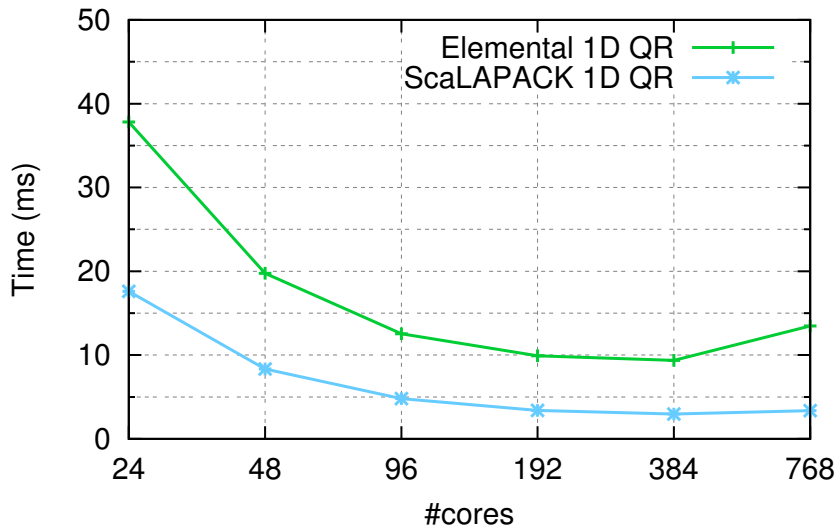
## Alternative Algorithms

- TSQR  $2nb^2$  flops, one QR reduction of size  $b^2/2$
- HhQR (and form  $T$ )  $3nb^2$  flops,  $2b$  reductions of size  $O(b)$
- Yamamoto's  $4nb^2$  flops, two QR reductions of size  $b^2/2$

For square matrices, flop costs of panel factorization are lower order:  $O(n^2b)$

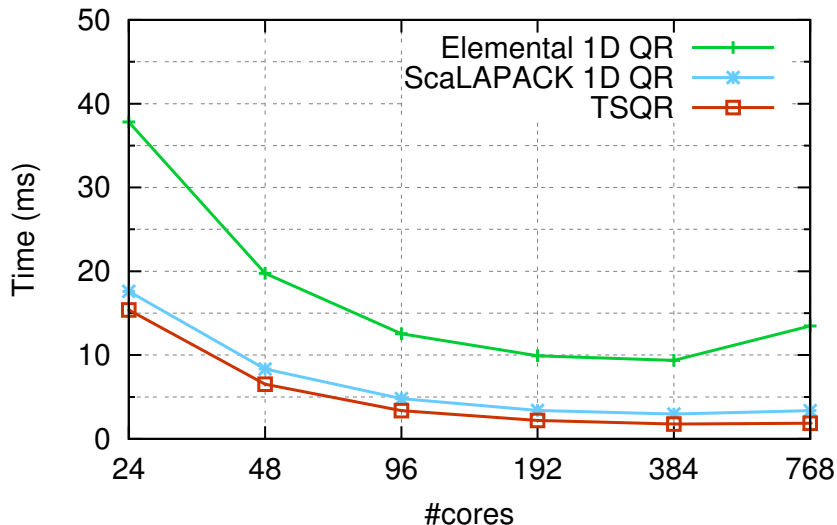
# Performance for Tall-Skinny Matrices

QR strong scaling on Hopper (122,880-by-32 matrix)



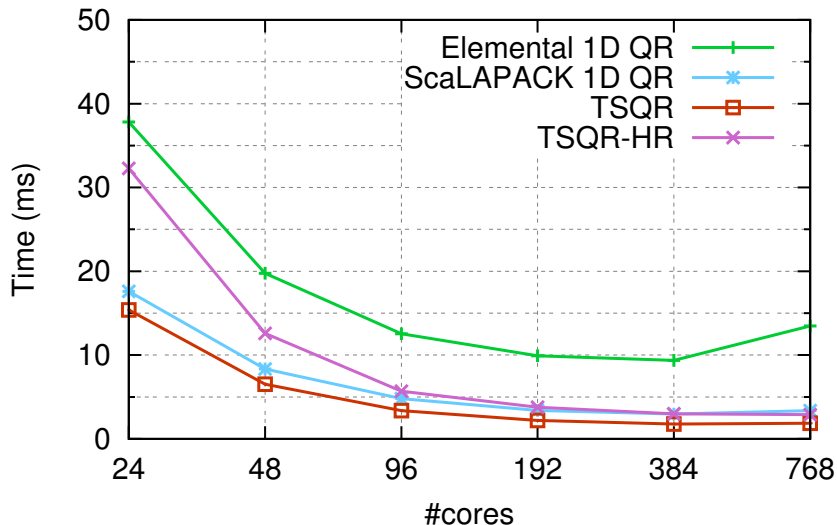
# Performance for Tall-Skinny Matrices

QR strong scaling on Hopper (122,880-by-32 matrix)



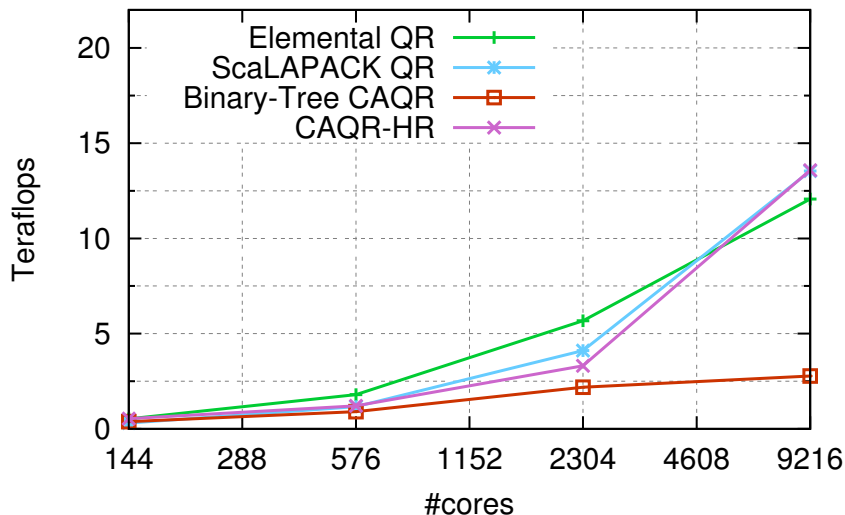
# Performance for Tall-Skinny Matrices

QR strong scaling on Hopper (122,880-by-32 matrix)

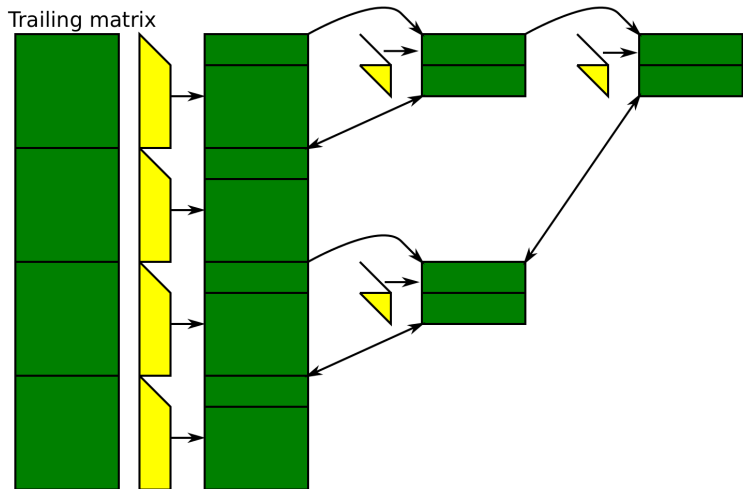


# Performance for Square Matrices

QR weak scaling on Hopper (15K-by-15K to 131K-by-131K)

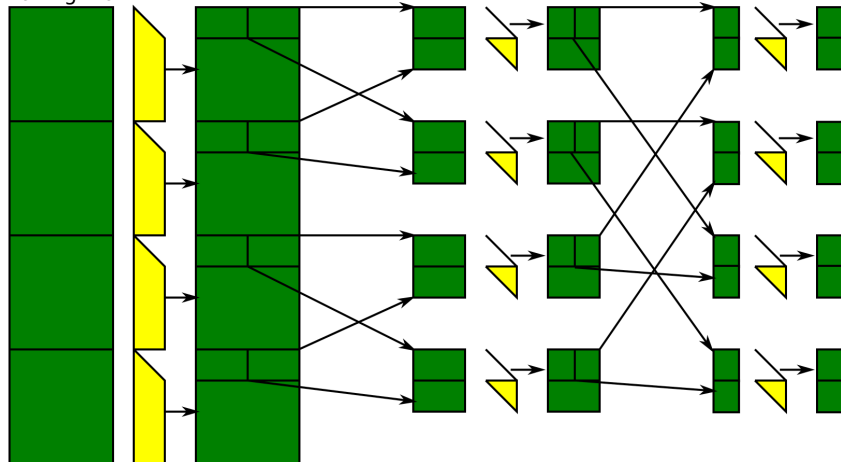


# Binary-Apply CAQR



# Scatter-Apply CAQR

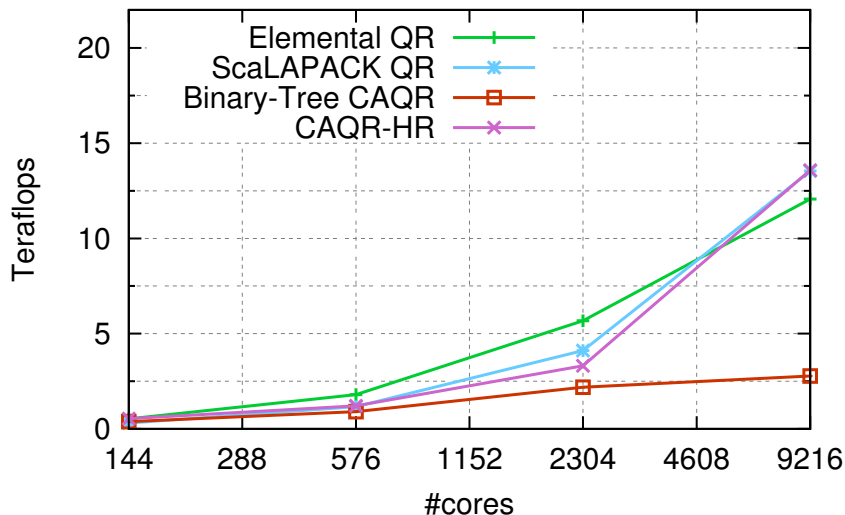
Trailing matrix



Similar to performing an all-reduce by reduce-scatter followed by all-gather

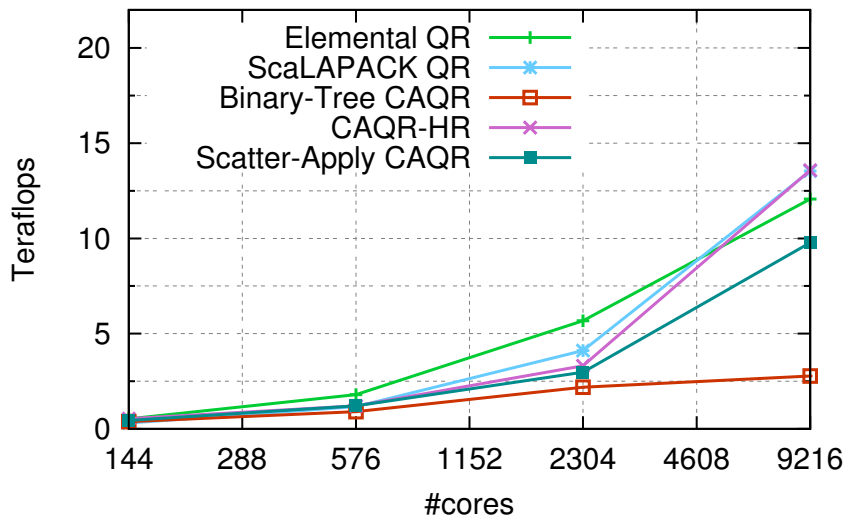
# Performance for Square Matrices

QR weak scaling on Hopper (15K-by-15K to 131K-by-131K)



# Performance for Square Matrices

QR weak scaling on Hopper (15K-by-15K to 131K-by-131K)

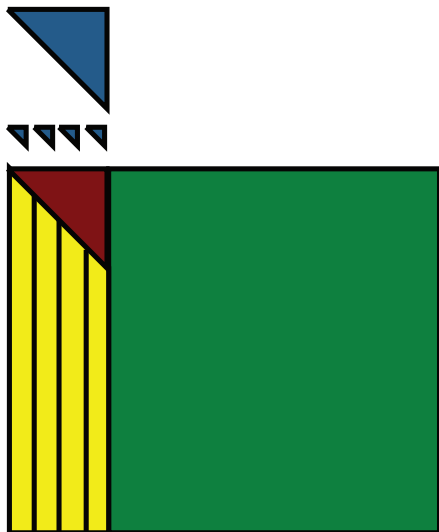


# Two-Level Aggregation

Block size trades off time spent in panel factorizations with efficiency of matrix multiplications

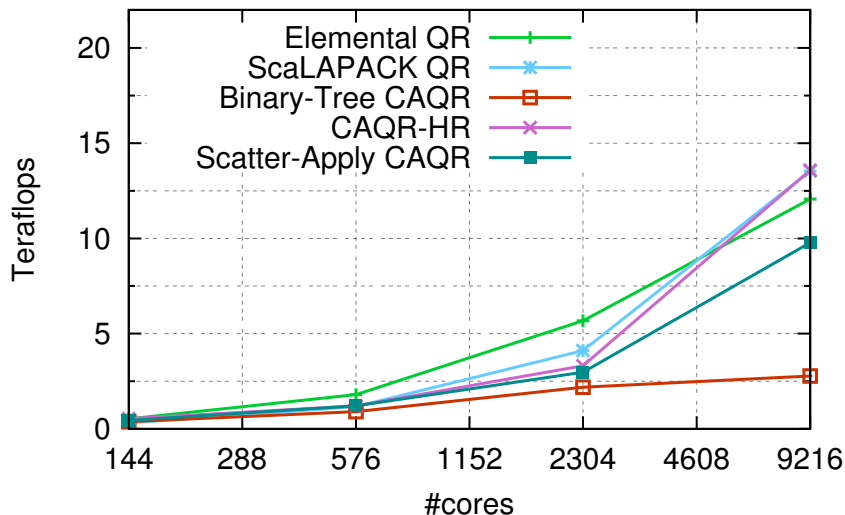
Solution:

- Use another level of compact WY blocking
- Allow for larger local matrix multiplications
- (Can't use with CAQR)



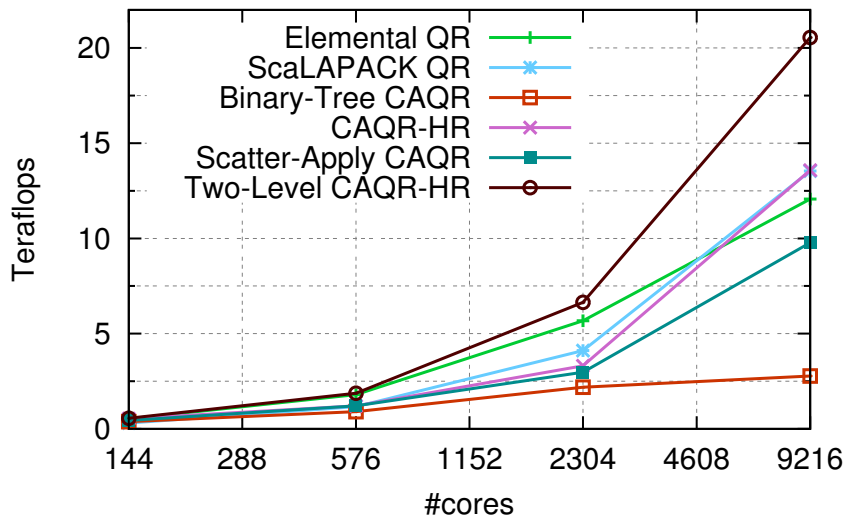
# Performance for Square Matrices

QR weak scaling on Hopper (15K-by-15K to 131K-by-131K)



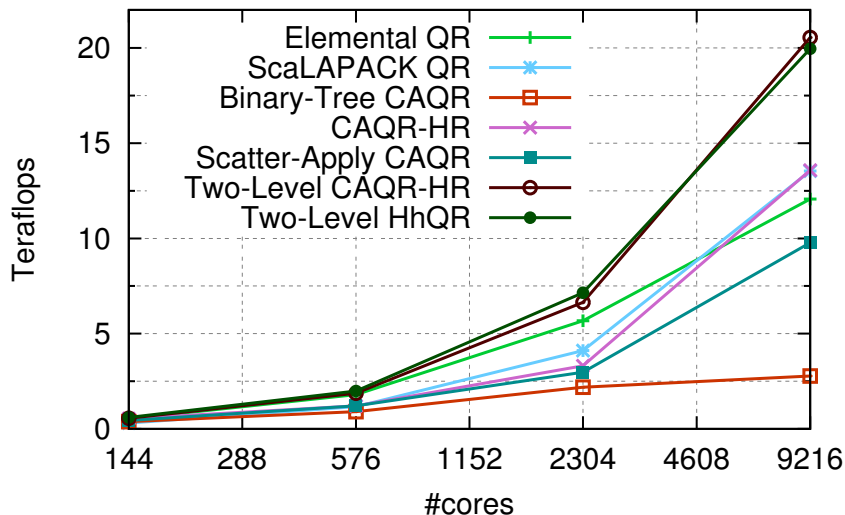
# Performance for Square Matrices

QR weak scaling on Hopper (15K-by-15K to 131K-by-131K)



# Performance for Square Matrices

QR weak scaling on Hopper (15K-by-15K to 131K-by-131K)



# Conclusions

- Householder reconstruction provides best of both worlds
  - latency-avoiding panel factorization
  - matrix multiplication trailing matrix updates
  - backwards compatibility for performance portability
- Scatter-apply technique improves CAQR trailing matrix update
- Two-level aggregation most important optimization on Hopper
- We expect Householder reconstruction to become more valuable as relative latency and synchronization costs increase

# Thanks!

For full details:

## **Reconstructing Householder Vectors from TSQR**

Grey Ballard, James Demmel, Laura Grigori,  
Mathias Jacquelin, Hong Diep Nguyen and Edgar Solomonik

[http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/  
EECS-2013-175.html](http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-175.html)

[gmballa@sandia.gov](mailto:gmballa@sandia.gov)

# References I



L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley.

*ScaLAPACK Users' Guide*.

SIAM, Philadelphia, PA, USA, May 1997.

Also available from <http://www.netlib.org/scalapack/>.



J. Demmel, L. Grigori, M. Hoemmen, and J. Langou.

Communication-optimal parallel and sequential QR and LU factorizations.

*SIAM Journal on Scientific Computing*, 34(1):A206–A239, 2012.



Jack Poulson, Bryan Marker, Robert A. van de Geijn, Jeff R. Hammond, and Nichols A. Romero.

Elemental: A new framework for distributed memory dense matrix computations.

*ACM Trans. Math. Softw.*, 39(2):13:1–13:24, February 2013.

## Theorem

Let  $\hat{R}$  be the computed upper triangular factor of  $m \times b$  matrix  $A$  obtained via the TSQR algorithm with  $p$  processors using a binary tree (assuming  $m/p \geq b$ ), and let  $\tilde{Q} = I - \tilde{Y}\tilde{T}\tilde{Y}_1^T$  and  $\tilde{R} = S\hat{R}$  where  $\tilde{Y}$ ,  $\tilde{T}$ , and  $S$  are the computed factors obtained from Householder reconstruction. Then

$$\|A - \tilde{Q}\tilde{R}\|_F \leq F_1(m, b, p, \epsilon)\|A\|_F$$

and

$$\|I - \tilde{Q}^T\tilde{Q}\|_F \leq F_2(m, b, p, \epsilon)$$

where  $F_1, F_2 = O\left((b^{3/2}(m/p) + b^{5/2}\log p + b^3)\epsilon\right)$  for  $b(m/p)\epsilon \ll 1$ .

# Numerical Experiments for Tall-Skinny Matrices

$\rho$	$\kappa$	$\ A - QR\ _2$	$\ I - Q^T Q\ _2$
1e-01	5.1e+02	2.2e-15	9.3e-15
1e-03	5.0e+04	2.2e-15	8.4e-15
1e-05	5.1e+06	2.3e-15	8.7e-15
1e-07	5.0e+08	2.4e-15	1.1e-14
1e-09	5.0e+10	2.3e-15	9.9e-15
1e-11	4.9e+12	2.5e-15	1.0e-14
1e-13	5.0e+14	2.2e-15	8.8e-15
1e-15	5.0e+15	2.4e-15	9.7e-15

Error of TSQR-HR on tall and skinny matrices ( $m = 1000, b = 200$ )

# Numerical Experiments for Square Matrices

Matrix type	$\kappa$	$\ A - QR\ _2$	$\ I - Q^T Q\ _2$
$A = 2 * \text{rand}(m) - 1$	$2.1e+03$	4.3e-15 (256)	2.8e-14 (2)
Golub-Klema-Stewart	$2.2e+20$	0.0e+00 (2)	0.0e+00 (2)
Break 1 distribution	$1.0e+09$	1.0e-14 (256)	2.8e-14 (2)
Break 9 distribution	$1.0e+09$	9.9e-15 (256)	2.9e-14 (2)
$U\Sigma V^T$ with exponential distribution	$4.2e+19$	2.0e-15 (256)	2.8e-14 (2)
The devil's stairs matrix	$2.3e+19$	2.4e-15 (256)	2.8e-14 (2)
KAHAN matrix, a trapezoidal matrix	$5.6e+56$	0.0e+00 (2)	0.0e+00 (2)
Matrix ARC130 from Matrix Market	$6.0e+10$	8.8e-19 (16)	2.1e-15 (2)
Matrix FS_541_1 from Matrix Market	$4.5e+03$	5.8e-16 (64)	1.8e-15 (256)
DERIV2: second derivative	$1.2e+06$	2.8e-15 (256)	4.6e-14 (2)
FOXGOOD: severely ill-posed problem	$5.7e+20$	2.4e-15 (256)	2.8e-14 (2)

Errors of CAQR-HR on square matrices ( $m = 1000$ ). The numbers in parentheses give the panel width yielding largest error.